第6章 Docker Compose

1 简介

Docker Compose是一个需要在Docker主机上进行安装的Docker容器编排外部工具。其并不是通过脚本或各种冗长的Docker命令来将应用组件组织起来，而是通过一个声明式的配置文件描述整个应用，然后通过一条命令完成应用部署。部署成功后，还可通过一系列简单命令实现对其完整生命周期的管理。

2 安装

Docker官网中的安装步骤非常详细，请按照官网安装。

https://docs.docker.com/compose/install/other/

其实就是通过下面的命令将docker-compose下载到本地。

```
curl -SL
https://github.com/docker/compose/releases/download/v2.13.0/docker-
compose-linux-x86_64 -o /usr/local/bin/docker-compose
```

然后再为该文件添加可执行权限即可。

```
chmod +x /usr/local/bin/docker-compose
```

3 compose文件

文件简介

Docker Compose使用YAML文件来定义服务。官方推荐的默认文件名为compose.yml，但同时也支持docker-compose.yml。不过也可以在运行docker-compose命令时通过-f选项来指定配置文件。

由于一个compose文件中定义的为一个项目的所有服务，所以一般为在创建compose文件之前先新建一个目录，目录名称一般为项目名称，然后再将项目所需的所有资源文件、微服务的Dockerfile放入该目录，并在该目录中新建compose文件。

compose文件中包含6个顶级属性：version、services、networks、volumes、configs与secrets，及很多的它们下面所包含的属性。下面简单介绍一下常用的属性。

version

version是一个顶级属性，但已经过时，不再需要在compose文件中出现了。

networks

networks作为一个顶级属性，用于定义和创建应用中所使用到的所有网络。其下包含的第一级属性即为网络名称，这个网络名称可以随意命名。而在网络名称下还可包含很多的属性，常用属性如下：

name

networks下的第一级属性—网络名称，并不是真正的网络名称，而仅仅是网络名称的一部分。在真正生成网络后，其真正的网络名称格式为：当前compose文件所在目录名_ networks下的第一级属性。

但如果设置了name属性，则网络名称即为这里指定的名称，不会出现名称再合成情况。

driver

用于指定网络驱动，缺省驱动为Bridge。

volumes

volumes作为一个顶级属性，用于定义和创建应用中所使用到的所有volume。其下包含的第一级属性即为volume的卷标，这个卷标可以随意命名。这个卷标所代表的是当前Docker主机中的目录，至于该目录的具体位置，是由系统自动分配的。

在网络名称下还可包含很多的属性，但这些属性并不常用，所以这里不进行介绍了。

serivces

services是一个顶级属性，用于定义一个应用中所包含的服务。Docker Compose会将每个服务部署在各自的容器中。其下包含的第一级的属性即为服务名称，这个名称可以根据服务内容随意命名。而在服务名称下还可包含很多的属性，常用属性如下：

build

用于指定一个Dockerfile的路径。而该Dockerfile则是用于创建当前服务镜像的。这个路径可以是以斜杠(/)开头的绝对路径，也可以是相对于当前compose文件的、以点(.)号开头的相对路径。

如果Dockerfile文件名不是默认名称，则需要通过build下的context属性指定路径，dockerfile属性指定文件名。

image

用户指定当前服务所需要使用的镜像，这个镜像可以是本地镜像，也可以是远程镜像仓库中的镜像(会自动pull)。

如果设置了build，此时再设置的image属性即为构建出的镜像的名称与Tag。

container_name

该属性用于设置容器名称，但并不是必须的。如果没有设置该属性，容器名称则会采用“合成方式”。而合成时需要用到services下的第一级属性。

在services下存在一级属性，称为服务名称。该级属性是作为services下的第一级属性出现的。服务名称将来会作为容器名称的一部分出现。容器的名称格式为：当前compose文件所在目录名_ 服务名称。

如果在services下没有指定image属性，而是使用bild属性，即没有现成的镜像，而是根据build下指定的Dockerfile生成镜像，此时生成的镜像名称格式为：当前compose文件所在目录名-服务名称。

ports

一个列表。前面为暴露出的端口号，后面为容器中应用的端口号。如果仅设置了一个端口号，那么这个端口号是容器中应用的端口号，其暴露到宿主机的端口号会被随机分配。

command

用于覆盖Dockerfile中的CMD指令内容，即启动该服务容器后立即运行的命令。如果直接按照Dockerfile中的CMD指令内容执行即可，则compose文件中无需该command属性。

depends_on

一个列表。用于指定当前服务的启动所依赖的应用名称。即列表中指定的服务会先于当前服务启动。

networks

用于指定当前服务容器要连接到的网络。该网络必须是已经存在的，或通过顶级属性networks创建的网络。

volumes

用于指定当前服务容器所使用到的所有volume。这些volume可以使用路径与卷标两种方式。

4 常用命令

Docker Compose通过docker-compose系列命令查看和控制compose中的所有服务容器。

docker-compose config

检查compose文件是否正确。可添加选项-q，表示只有存在问题时才有输出。

docker-compose up

启动compose中的所有容器。-d选项表示后台启动。

docker-compose stop

停止compose中所有服务容器或指定服务容器。通过在命令后添加服务名称来指定。

docker-compose restart

重启compose中所有服务容器或指定服务容器。通过在命令后添加服务名称来指定。

docker-compose start

启动compose中所有服务容器或指定服务容器。通过在命令后添加服务名称来指定。

docker-compose kill

通过发送SIGKILL信号强制停止指定服务的容器。

docker-compose stop

优雅停止compose中所有服务容器或指定服务容器。通过在命令后添加服务名称来指定。

docker-compose rm

删除compose中处于停止状态的所有服务容器或指定服务容器。通过在命令后添加服务名称来指定。

5 项目构建

项目代码

准备一个项目，该项目需要连接MySQL与Redis。

定义Dockerfile

在Docker主机的任意目录中新建一个目录，该新建目录名称与项目名称相同。例如，这里在/root目录中mkdir一个名称为ssrm目录。然后在该目录中新建Dockerfile文件。

```
FROM openjdk:8u102
LABEL num="20150018" name="zhangsan" class="20-yun-1"  # 请使用自己的个人信息
COPY ssrm-0.0.1-SNAPSHOT.jar ssrm.jar
ENTRYPOINT ["java", "-jar", "ssrm.jar"]
EXPOSE 9000
```

项目打包

将项目package打为jar包，然后再上传到Docker主机的/root/ssrm目录中。

6 Compose编排启动项目

定义compose.yml

在ssrm目录中新建一个文件compose.yml，文件内容如下：

```
services:
  app:  # 请使用自己姓名拼音的首字母
    build: ./
    container_name: myapp    # 请使用自己姓名全拼
    ports:
      - 9000:8080
    volumes:
      - ./logs:/var/applogs
    depends_on:
      - mysql
      - redis

  mysql:
    image: mysql:5.7
    environment:
      MYSQL_ROOT_PASSWORD: 111
    ports:
      - 3306:3306
    volumes:
      - /root/mysql/log:/var/log/mysql
      - /root/mysql/data:/var/log/mysql
      - /root/mysql/conf:/etc/mysql/conf.d

  redis:
    image: redis:7.0
    ports:
      - 6379:6379
    volumes:
      - /root/redis/redis.conf:/etc/redis/redis.conf
      - /root/redis/data:/data
    command: redis-server /etc/redis/redis.conf
```

> *注意，在项目的application.yml文件中，MySQL与Redis的主机地址要写为compose.yml文件中MySQL与Redis服务的服务名。*

启动所有容器

```
docker-compose config -q
docker-compose up -d
```